云计算环境中面向 DAG 任务的多目标调度算法 *

徐健锐 [1,2]，朱会娟 [3]

(1. 江苏大学 计算机科学与通信工程学院，江苏 镇江 212013; 2. 江苏联合职业技术学院 镇江分院，江苏 镇江 212016; 3. 中国科学院大学 计算机与控制学院，北京 100049)

摘 要：为了实现任务执行效率与执行代价的同步优化，提出了一种云计算环境中的 DAG 任务多目标调度优化算法。算法将多目标最优化问题以满足 Pareto 最优的均衡最优解集合的形式进行建模，以启发式方式对模型进行求解；同时，为了衡量多目标均衡解的质量，设计了基于 hypervolume 方法的评估机制，从而可以得到相互冲突目标间的均衡调度解。通过配置云环境与三种人工合成工作流和两种现实科学工作流的仿真实验测试，结果表明，比较同类单目标算法和多目标启发式算法，算法不仅求解质量更高，而且解的均衡度更好，更加符合现实云的资源使用特征与工作流调度模式。

关键词：云计算；工作流调度；多目标优化；Pareto 边界；亚马逊弹性计算云

中图分类号：TP301.6　　doi: 10.3969/j.issn.1001-3695.2017.07.0683

Multi-objective scheduling algorithm of DAG tasks in cloud computing

Xu Jianrui[1, 2], Zhu Huijuan[3]

(1. *School of Computer Science & Telecommunication Engineering Jiangsu University*, *Zhenjiang Jiangsu* 212013, *China*; 2. *Zhenjiang Branch Jiangsu Union Technical Institute*, *Zhenjiang Jiangsu* 212016, *China*; 3. *School of Computer & Control Engineering*, *University of Chinese Academy of Sciences*, *Beijing* 100049, *China*)

Abstract: For implementing the synchronization optimization of tasks execution efficiency and execution cost, a multi-objective scheduling optimization algorithm of DAG tasks in cloud environment is presented. Our algorithm defines the multi-objective optimization problem as the trade-off optimal solutions set satisfying Pareto optimal and solves this model by the heuristic method. At the same time, for evaluating the quality of multi-objective trade-off solutions, a evaluation mechanism based on hypervolume method is designed, which can obtain the trade-off scheduling solutions with conflict objectives. Through setting cloud environment and three kinds of synthetic workflow and two kinds of real-world scientific workflow, we construct some simulation experiments. The results show that, compared with the same type of single objective algorithm and multi-objective heuristic algorithm, our algorithm not only has higher solving quality, but has better trade-off degree of solutions, which can conform to the mode of resource utility and workflow scheduling in real-world cloud.

Key Words: cloud computing; workflow scheduling; multi-objective optimization; Pareto front; Amazon EC2

0 引言

科学领域的实验任务通常定义为工作流形式，其任务根据数据流与计算相关性形成链式结构，如计算密集型和数据密集型工作流应用，其数据量和计算需求量巨大，需要高性能计算环境支持运行[1]。云计算的出现为科学工作流的执行提供了更高效的技术与支撑环境[2]。云以虚拟机（virtual machines, VM）的形式提供计算资源，而工作流中任务与计算资源间的映射问题即为工作流调度问题。云工作流调度包括两个层次：一是任务与 VM 间的映射，二是在单个 VM 上任务的顺序执行[3]。本文将以 Amazon EC2 云环境为基础解决云工作流调度的多目标最优化问题，重点关注于工作流调度执行跨度与执行代价的最优化，寻找最优调度方案。

相关研究中，文献[4]提出一种异构预算约束调度算法 HBCS，通过定义代价因子调整可用预算与最低价格的比例，实现优化调度。文献[5]提出一种异构最快完成时间调度算法 HEFT，通过赋予任务不同优先级，最小化任务调度时间。文献[6]提出一种基于预算约束的异构最快完成时间算法 BHEFT，该算法是对 HEFT 算法的扩展实现，考虑了任务调度的最优预算约束问题。文献[7]提出基于离散粒子群优化的工作流调度算

基金项目：国家自然科学基金项目（批准号：61302124）；江苏省高校自然科学研究面上项目（批准号：16KJB520010）

作者简介：徐健锐（1975-），男，江苏扬州人，副教授，硕士，主要研究方向为复杂网络预测与评价、云计算与大数据分析、教育信息化及职业教育人才培养模式（xujr75@126.com）；朱会娟（1984-），女，河南洛阳人，博士研究生，主要研究方向为数据清洗、数据融合、数据分析.

法 CWDPSO，算法以资源使用成本和安全满意度为优化目标，利用粒子群进化求解了资源分配方案。HEFT、BHEFT、HBCS 和 CWDPSO 四种算法均是将求解目标转换为单目标最优化，并不适用于 Amazon EC2 的云工作流调度场景。

多目标调度中，文献[8]提出 POSH 算法，将执行跨度与执行代价同步考虑，同时以用户定义的偏好因子设置两者权重，实现多目标优化。文献[9]中的 NSPSO 算法和文献[10]中的 Fussy-PSO 算法均尝试使用粒子群优化算法实现工作流调度时执行跨度与执行代价的均衡。文献[11]提出多目标异构最快完成时间算法 MOHEFT，旨在通过 Pareto 线性启发式算法对 HEFT 进行扩展，实现多目标优化。然而，以上多目标调度算法仅适用于传统异构计算环境，IaaS 云环境在计算模式、数据以及定价模型上均不同于传统计算环境。

本文将提出一种云环境中的工作流多目标调度算法 CMOHEFT。算法以工作流任务的执行跨度 makespan 和执行经济代价的同步最小化为目标，通过启发式方法得到了多目标间的均衡最优解集合，并以 Pareto 边界和 hypervolume 值的方式对均衡解进行了有效性评估。

1　Amazon EC2 云环境

Amazon 弹性计算云 EC2 是一种 IaaS（基础设施即服务，infrastructure as a service）云服务，可开放地向用户提供 Amazon 的计算设施。“弹性”特征表明用户可以通过请求或释放资源和实例的方式扩展和收缩自身的基础设施。目前，Amazon EC2 提供三种类型的实例：

a）预留型实例，允许用户对多个主机资源作长期预订。

b)按需型实例，允许用户仅按请求的资源和请求时间进行付费，付费单位为小时。

c)Spot 型实例，允许用户对未使用的 Amazon 资源进行投标，用户指定资源使用的最大出价，若出价高于当前 spot 价格（取决于资源需求量），则用户获得实例；若出价小于 spot 价格，则 Amazon 回收实例。

Amazon EC2 提供 14 种拥有不同性能和代价的不同类型按需实例。实例的计算性能根据弹性计算单元 ECU（Elastic Compute Unit）进行定义，等同于一个拥有 1.0-1.2GHz 的 Xeon 处理器的能力。表 1 总结了这些资源的平均性能、单位小时计算能力的代价及单位费用的资源每秒浮点运算数 GFLOPs（millions of floating point operation per second）。本文的多目标工作流调度算法将以表 1 的 5 种类型实例进行性能评估。

表 1　Amazon EC2 实例类型

实例类型	平均性能/GFLOPS	价格/$/h	GFLOPS/$
m1.small	2.0	0.1	19.6
m1.large	7.1	0.4	17.9
m1.xlarge	11.4	0.8	14.2
c1.medium	3.9	0.2	19.6
c2.xlarge	50.0	0.8	62.5

2　模型描述

2.1　工作流 DAG 任务模型

定义工作流应用为有向无循环图 DAG，$W=(A,D)$，其中，A 表示包括 n 个任务的任务集，$A=\cup n\ i=1\{A_i\}$，D 表示任务间的控制流和数据流依赖关系，$D=\{(A_i,A_j,Data_{ij})|(A_i,A_j)\in A\times A\}$，$Data_{ij}$ 表示任务 A_i 与 A_j 间的数据传输量。令 $pred(A_i)=\{A_k|(A_k,A_i,Data_{ki})\in D\}$表示任务 A_i 的前驱任务集，即必须在 A_i 开始之前完成。假设每个任务 A_i 的计算负载是已知的，且表示为可执行的机器指令数量。

2.2　资源模型

假设云平台由 m 个异构资源集合组成，$R=\cup m\ j=1\{R_j\}$，由 Amazon EC2 的云资源类型提供，如：m1.small、m1.medium、m1.large、m1.xlarge、c1.medium 及 c1.xlarge。对于确定类型的给定资源 R_j，其平均性能由 GFLOPs 进行度量，即：Floating-point Operations Per Second，每秒所执行的浮点运算次数。在本文的工作流模型中，假设工作流任务可以并行执行的方式运行于如表 1 所示的 Amazon EC2 虚拟机实例上，资源使用以单位小时收取费用（表 1 第三列），最终的用户付费成本不仅取决于资源使用，而且包括数据存储与不同 VM 实例间的数据传输，具体包括以下四个部分：每小时的资源使用价格 PE_{Ri}；单位 MB 的数据存储价格 PS_{Ri}；单位 MB 的数据接收价格 PI_{Ri}；单位 MB 的数据发送价格 PO_{Ri}。

2.3　问题定义

本文的工作流调度问题即为将工作流任务调度至 Amazon 云资源上执行，使得执行时间 makespan 和经济代价达到最小。令 $sched(A_i)$表示任务 A_i 调度的执行资源，以下分别定义两个优化目标：

1）执行时间 makespan

令 *execution time* $t(A_i,R_j)$表示任务 A_i 在资源 $R_j=sched(A_i)$的计算时间与接收来自任意前驱任务 $A_p\in pred(A_i)$的最大输入数据时间之和，表示为

$$t(A_i,R_j)=\max_{A_p\in pred(A_i)}\frac{Data_{pi}}{b_{pj}}+\frac{workload(A_i)}{s_j} \tag{1}$$

其中:$Data_{pi}$ 表示任务 A_p 与 A_i 间的数据传输量，b_{pj} 表示任务 A_p 的执行资源与资源 R_j 间的带宽，$workload(A_i)$表示任务 A_i 的指令长度，s_j 表示资源 R_j 的计算速度，单位为每秒执行的机器指令数 MIPS。任务 A_i 的完成时间 T_{Ai} 包括任务本身的执行时间和其前驱的执行时间，表示为

$$T_{A_i}=\begin{cases} t(A_i,sched(A_i)), pred(A_i)=\varnothing \\ \max_{A_p\in pred(A_i)}\{T_{A_p}+t(A_i,sched(A_i))\}, \\ pred(A_i)\neq\varnothing \end{cases} \tag{2}$$

工作流的执行时间 makespan 定义为所有任务中的最大完成时间为

$$T_W=\max_{i\in[1,n]}T(A_i,sched(A_i)) \tag{3}$$

2）经济代价

工作流执行的经济代价包括两个部分：计算代价 $C^{(comp)}$和数据传输与存储代价 $C^{(data)}$。

令 $C(data)$ (Ai,Rj)表示任务 A_i 在资源 R_j 上执行过程中传输输入数据 $In(A_i)$、输出数据 $Out(A_i)$及存储数据 $Data(A_i)$的代价，表示为

$$C_{(A_i,R_j)}^{(data)} = Data(A_i) \times t(A_i,R_j) \times PS_{R_i} + In(A_i) \times PI_{R_i} + Out(A_i) \times PO_{R_i} \tag{4}$$

令 $C(comp)$ Rj 表示使用资源 R_j 的代价，对于执行 A_i 的 R_j，令 $t(start)$ Ai 表示任务的开始时间，$t(end)$ Ai 表示任务的结束时间，可表示为

$$t_{A_i}^{(start)} + t(A_i,R_j) + \max_{A_i \in pred(A_p)} \frac{Data_{ip}}{b_{jp}} \tag{5}$$

同时，假设传输输入数据 $In(A_i)$和输出数据 $Out(A_i)$的时间均处于 $t(start)$ Ai 和 $t(end)$ Ai 之间。

考虑在资源 R_j 上调度的 p 个任务集合为$\{J_1,J_2,\ldots,J_p\}$，$p<n$ 且 $sched(J_i)=R_j,i\in[1,p]$，任务基于开始时间进行排列：$t(start)$ $J1<$ $t(start)$ $J2<\ldots<t(start)$ Jp。基于以上排列，将所有任务划分为 $q(q\leq p)$个不同的群组 $G(j)$ k，$1\leq k\leq q$，使得同一组内的所有任务可以在无须释放资源的同时连续执行。群组中拥有最大开始时间的任务完成后，资源即可释放。

通过以下三个规则，构造第一个群组 $G(j)$ $1=\{J_1,J_2,\ldots,J_r\}$，$r\leq p$：

规则 1　第 1 个任务 J_1 属于第一个群组：$J_1\in G(j)$ 1；

规则 2　每个任务 $J_i\in G(j)$ $1(2\leq i\leq r)$在资源释放前完成。这表明由于 J_{i-1} 的执行，资源仍被租用时 Ji 即可开始：

$$t_{J_i}^{(start)} < t_{J_1}^{(start)} + \left\lceil \frac{t_{J_{i-1}}^{(start)} - t_{J_1}^{(start)}}{3600} \right\rceil \times 3600 \tag{6}$$

规则 3　下一任务 $J_{r+1}\notin G(j)$ $1(r+1\leq p)$在资源已经释放时，在开始时间 $t(start)$ $Jr+1$ 时立即开始执行，即：任务 J_r 完成执行后，执行 J_r 的最后租用周期 1 小时已经截止，而且资源 R_j 在 t(end) Jr 与 $t(start)$ $Jr+1$ 间是为空闲，表示为

$$t_{J_1}^{(start)} \left\lceil \frac{t_{J_r}^{(end)} - t_{J_1}^{(start)}}{3600} \right\rceil \times 3600 < t_{J_{r+1}}^{(start)} \tag{7}$$

直到最后的任务 J_p 分配至一个群组后，即可建立连续的群组。第二个群组 $G(j)$ 2 以 J_{r+1} 代替 J_1 以同样的方式构建，同样的策略被使用至剩余群组的构建中。建立所有群组后，定义使用资源 R_j 的代价 $C(comp)$ Rj 为执行所有群组需要的小时数与单位小时代价之积：

$$C_{R_j}^{(comp)} = PE_{R_j} \times \sum_{k=1}^{q} \left\lceil \frac{\sum_{A_i \in G_{kj}^{(k)}} t(A_i,R_j)}{3600} \right\rceil \tag{8}$$

执行工作流 $W=(A,D)$的经济代价为所有 m 个资源的计算代价与数据传输与存储代价之和：

$$C_W = \sum_{j=1}^{m} C_{R_j}^{(comp)} + \sum_{(A_i,A_j,Data_{ij})\in D} C_{(A_i,R_j)}^{(data)} \tag{9}$$

3　多目标优化

多目标最优化问题通常可定义为：寻找使得矢量函数 $f(x)=[f_1(x),f_2(x),\ldots,f_o(x)]$最小化的所有解 x，其中，o 表示优化目标数量（本文为两个目标：执行时间 makespan 和经济代价）。工作流调度问题中每个解 x 包括两个矢量：第一个矢量为包括 n 个元素的矢量$(x_1,x_2,\ldots,x_n)$，n 表示工作流的任务数量($n=|A|$)，$x_i=sched(A_i)$表示任务 $A_i\in A$ 执行的资源；第二个矢量为大小为 n 的排列 p，表示所有任务执行的顺序。

对于工作流调度问题，同时找到实现执行时间 makespan 和经济代价的最小化的解是不可能的。为了解决这一问题，本文引入 Pareto 占优的概念对同步最优进行评估。

如果解 y 的 makespan 和代价均小于解 z，则称解 y 占优解 z。相反，如果两个解之间不存在相互占优（一个解拥有更小 makespan，而另一个解拥有更小代价），则称两个解为非占优。以图 1 为例，解 a 占优解 b，由于其拥有更好的 makespan 和代价。同样地，解 a 占优解 c。同时，解 a 与解 d 是非占优的，由于解 a 的 makespan 更优，而解 d 的代价更优。最优非占优解的集合称为 Pareto 边界（包含解 a、d 和 e 的趋势线），表示不同目标间的均衡解集合。该集合中的每个解表示拥有不同 makespan 和代价的工作流任务的不同映射方案。

度量均衡解质量的方法为超体积 hypervolume。给定均衡解 X 的集合，hypervolume $HV(X)$测量的是 X 中的点与参考点 W 间的封闭区域，如图 1 所示，通常选择为最大目标值（即最高的 makespan 和经济代价）。因此，在 X 中的点越多不同越好，其 $HV(X)$值也越高。图 1 中，包含实线图周围点的解集合优于包含正方形中点集合的解，因此，包含所有实线图周围点的集合的 hypervolume 值高于包含正方形中点集合的 hypervolume 值。

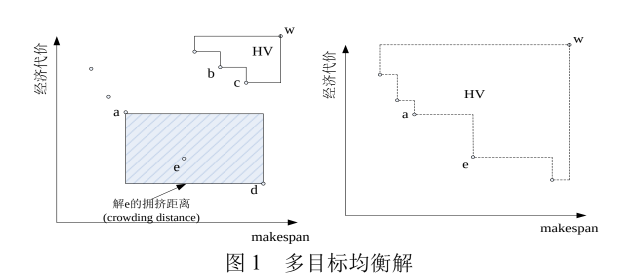


图 1　多目标均衡解

4　多目标工作流调度算法设计

本节笔者改进了传统异构计算环境中的单目标优化算法 HEFT 和多目标优化 MOHEFT，使其可以适用于 Amazon EC2 的云环境中。传统异构系统中，资源数量与类型均是已知的，然而，云计算环境中的资源总量虽是有限的，但其资源提供模式是按需和动态的，资源可用性及其计算能力均是动态变化的。

为了使算法适应于新的云环境，考虑资源输入量为 $m=N\times I$，其中，I 表示云提供者提供的实例类型数量，通过这种设置，可以确保资源最大量 N 下的所有组合均是可能的。

算法 1 为单目标算法 CHEFT 的执行过程，算法的设计思

路是：将任务调度过程划分为两个阶段，任务分级阶段和任务映射阶段。任务分级阶段中利用自顶向下分级的方式计算任务至工作流结构的出口任务间的距离（以任务至出口任务之间所有路径上边的最大数量表示），然后根据各个任务的分级值对任务进行降序排列，得到任务调度优先序列，通过该分级方法可以使得拥有更大依赖性的任务优先得到执行。任务映射阶段中即在所有资源上寻找 makespan 最小的调度解。在该过程中，为了符合云环境的调度需求，需求测试局部调度方案是否需求多于 N 个资源，如果不多于 N，则通过公式 $T_{Ranki}\leftarrow\max_{Ap\in pred(Ranki)}\{T_{Ap}+t(Rank_i,R_j)\}$ 计算 makespan，否则，该解的 makespan 设置为无穷。

算法 1 CHEFT

Require: $W=(A,D), A=\cup\, n\ i=1\{A_i\}$//工作流应用

Require: N//最大同步实例（资源）数量

Require: I//不同实例类型数量

Require: $R=\cup\, m\ j=1\{R_j\}$//资源集合，$m=N\times I$

Ensure: $sched_W=\{(A_i,sched(A_i))|A_i\in A\}$//工作流调度

function CHEFT(W,R)

Rank←B-RANK(A) //根据 B-rank 对任务排序

$sched_W\leftarrow\varnothing$//初始化工作流调度为空集

for $i\leftarrow1,n$ do//迭代所有分级任务

$T_{min}\leftarrow\infty$

for $j\leftarrow1,m$ do//迭代所有资源

if COUNTRESOURCES($sched_W,m$)$\leq N$ then//如果使用实例数小于 N

$T_{Ranki}\leftarrow\max_{Ap\in pred(Ranki)}\{T_{Ap}+t(Rank_i,R_j)\}$//计算 $Rank_i$ 的完成时间

else

$T_{Ranki}\leftarrow\infty$//标识调度为无效

end if

if $T_{Ranki}<T_{min}$ then//保存最小完成时间

$T_{min}\leftarrow T_{Ranki}$

$R_{min}\leftarrow R_j$

end if

end for

$sched_W\leftarrow sched_W\cup(Rank_i,R_{min})$//调度任务

end for

return $sched_W$

end function

CHEFT 与传统单目标算法不同之处在于算法 1 的步骤 12～16 行，传统单目标算法在每次新的局部调度中计算 makespan，而 CHEFT 则测试局部调度方案是否需求多于 N 个资源，如果不多于 N，则计算 makespan（行 13），否则，该解的 makespan 设置为无限，即步骤 15。通过该方法，无效调度方案将无法验证行 17 中的条件，且将被丢弃并而保留需求资源小于 N 个同步实例的调度方案。

算法 2 为多目标算法 CMOHEFT 的执行过程，算法的设计思路是：首先，以自顶向下分级的方式计算任务分级值，然后，设置可容纳 K 个多目标均衡的空集 S。迭代访问根据分级值降序排列的任务集合，得到执行顺序。算法的目标在 m 个可用资源上是不断扩展集合 S，以创建并存储新的映射解。新的映射的搜索过程为寻找到 K 个解为止。最后，以 Pareto 占优及 hypervolume 法评估解的质量，得到多目标最优解。

算法 2 CMOHEFT

Require: $W=(A,D), A=\cup\, n\ i=1\{A_i\}$//工作流应用

Require: N//最大同步实例（资源）数量

Require: I//不同实例类型数量

Require: $R=\cup\, m\ j=1\{R_j\}$//资源集合，$m=N\times I$

Require: K//均衡解数量

Ensure: $S=\cup\, K\ i=1\{sched_W\}, sched_W=\{(A_i,sched(A_i))|A_i\in A\}$//K 个均衡调度解集合

function CMOHEFT(W,R,K)

$Rank$←B-RANK(A) //根据 B-rank 对任务排序

for $k\leftarrow1,K$ do//创建 K 个空的工作流调度解

$S_k\leftarrow\varnothing$

end for

for $i\leftarrow1,n$ do//迭代所有分级任务

$S'\leftarrow\varnothing$

for $j\leftarrow1,m$ do//迭代所有资源

for $k\leftarrow1,K$ do//迭代所有均衡调度解

$s\leftarrow S_k\cup(Rank_i,R_j)$ //扩展所有中间调度方案

if COUNTRESOURCES($sched_W,m$)$>N$ then//如果使用实例数多于 N

$T_s\leftarrow\infty$//标记调度为无效

$C_s\leftarrow\infty$

end if

$S'\leftarrow S'\cup\{s\}$//添加新映射至所有中间调度

end for

end for

S'←SORTCROWDDIST(S',K)//根据拥挤距离排序

S←FIRST(S',K)//选择 K 个最高拥挤距离的调度

end for

return S

end function

CMOHEFT 与传统多目标算法的不同之处在于算法 2 的 17～20 行，CMOHEFT 的资源量约束在第 17 行检查，如果未违背资源量约束，makespan 和代价按算法前文进行计算，如果违背，则设置为无限，这可以使算法丢弃第 24 行中的解，仅仅产生至多 N 个同步资源的均衡解。

算法时间复杂度分析。给定包括 n 个任务和 m 个资源的工作流调度场景，CHEFT 算法需要先计算每个任务的分级值，该

过程的时间复杂度为 $O(n)$。然后，需要将排序后的 n 个任务迭代方式在 m 个可用资源上寻找执行跨度 makespan 最小的调度解，该过程的时间复杂度为 $O(n\times m)$。因此，CHEFT 算法的时间复杂度为 $O(n)+O(n\times m)=O(n\times m)$。

CMOHEFT 算法与 CHEFT 算法的过程有两个不同之处：一是算法在每次迭代中多个解的创建方式，二是考虑资源提供均衡解的可能性，两处不同均体现在 CMOHEFT 算法中的 15～21 行中。考虑均衡解的集合大小为 K，通常情况下，均衡解的数量为常数，且其值小于 n 和 m，而 CMOHEFT 算法比较 CHEFT 仅额外需要执行 K 次迭代，因此，CMOHEFT 算法的最差时间复杂度为 $O(n\times m\times K)$。

5　实验配置

本节通过仿真实验以评估多目标算法 CMOHEFT 的有效性，仿真平台选择 WorkFlowSim[12]。首先，给出了评估算法性能质量的比较指标，然后，描述了测试工作流的不同类型和实验中 Amazon EC2 设施的配置情况。实验中除了实现 CHEFT 和 CMOHEFT 算法之外，还选择了 POSH 算法[8]作为性能比较的参考算法，选择的原因是该算法同为多目标优化算法，且也考虑了均衡解的问题。

5.1　评估指标

考虑三个指标比较三种算法的性能：执行工作流的最短 makespan、执行工作流的经济代价及 hypervolume 评价指标。前两个指标主要用来评价单个指标的优化性能，hypervolume 则是评价多目标最优均衡解的质量。

Hypervolume 的具体的计算方法是：给定由 CMOHEFT 算法计算得到的调度均衡解 X 的集合，定义 $HV(X)$表示一种 hypervolume 值，测量范围为是调度解集合 X 中的单个点（表示一个调度解）与所选择的参考点 W 间的封闭区域，基于工作流调度目标是执行 makespan 与执行代价的同步优化，参考点选取为调度均衡解集合 X 中的最高的 makespan 和执行代价 cost 所表示的点。然后，将解集合中的个体解按照其在第一维目标上的值进行降序排列，按照以下公式计算个体解 p_i 的独立占优区域的 Hypervolume 指标：

$$HV(p_i)=|\text{makespan}(p_{i-1})-\text{makespan}(p_i)|\times|\text{cost}(p_{i+1})-\text{cost}(p_i)| \quad (10)$$

其中，$2\le i\le K-1$，K 表示解集合中的解数量，makespan(p_i)表示解集中第 i 个个体解在执行跨度 makespan 上的值，cost(p_i)同理。

5.2　工作流应用

实验中引入两种类型的工作流应用进行测试：人工配置不同属性的多类型的人工合成工作流应用和现实科学领域中的现实工作流。

1）人工合成工作流

利用随机工作流产生器产生三种类型合成工作流（如图 2(a)和图 2(b)），主要目的是分析任务数量对调度结果的影响，因此，所定义的工作流类型可以覆盖多种类型的工作流结构：

Type-1 工作流：任务可以并行度为 1 和 2 时进行执行；

Type-2 工作流：任务执行的并行度较高，且工作流较均衡（同一层次上的任务数相同）；

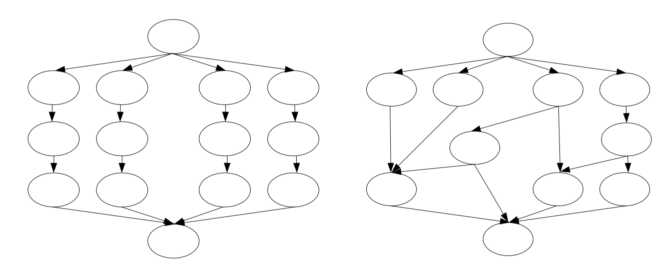
Type-3 工作流：任务执行的并行度较高，但工作流不均衡（同一层次上的任务数不同）。

每个任务的大小和数据量的产生符合高斯分布，对于每种类型工作流，任务分布于 100 与 1000 之间，实例数为 100。

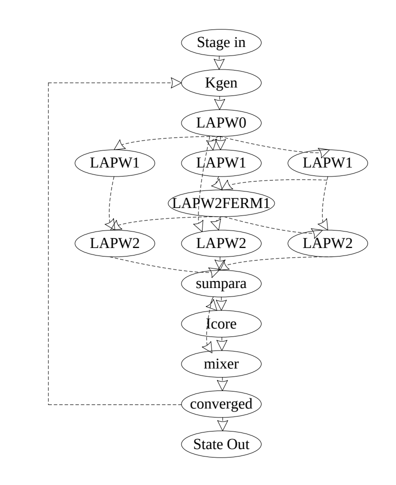
2）现实工作流应用

考虑两种现实工作流应用：WIEN2K 和 POV-Ray，结构分别如图 2(c)和图 2(d)。

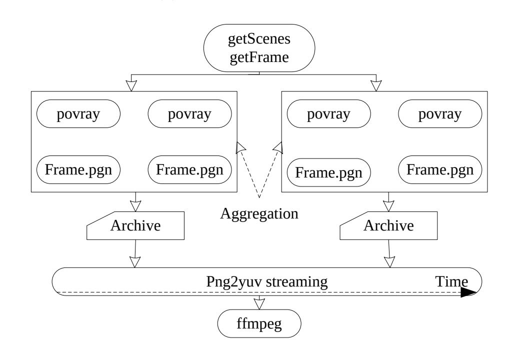
WIEN2K 是基于波纹方法并利用密度泛函理论计算固体电子结构的一种材料科学工作流应用，属于 Type-2 工作流类型，由连续同步任务执行的两个并行部分组成。POV-Ray 是一种构建三维图形的开放工具，多用于生物学、医学、建筑学和数学可视化领域中，也属于 Type-2 工作流类型。



(a)人工合成工作流结构-均衡　(b)人工合成工作流结构-非均衡



(c)WIEN2k 工作流结构



(d)POV-Ray 工作流结构

图 2　工作流应用结构

5.3　资源配置

令用户可访问的 Amazon 实例的最大数量为 N=20，实例共

有五种类型，如表 1 所示，即：I=5，m=N×I=20×5=100。同时，设 Amazon 至外部互联网间的输出数据量为常数，且该数据输出仅发生于工作流执行的结束，因此不会影响调度结果。仿真实验中，数据发送与接收的代价均为 0，即 PI_{Ri}=0，PO_{Ri}=0。

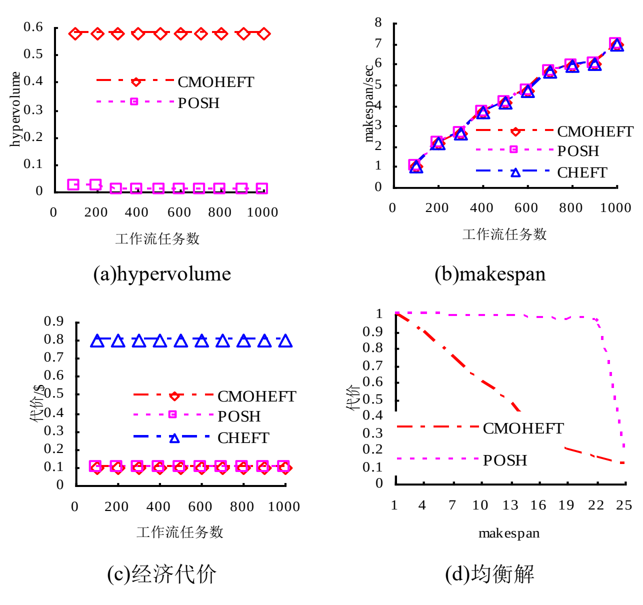
6 实验结果评估

6.1 合成工作流

1)Type-1 工作流

图 3 是三种算法在 makespan、经济代价和 hypervolume 性能上的结果。图 3(a)表明 CMOHEFT 在所有评估实例上的 hypervolume 均优于 POSH。图中未包含 CHEFT 是由于该算法仅得到最优 makespan 的单一解。明显地，对于 Type-1 工作流，CMOHEFT 在计算调度解时始终拥有相同的 hypervolume 值，这表明均衡解的最优集合的形态并未随着工作流规模的变化而发生变化。在 makespan 方面（图 3(b)），三种算法得到了相同解，这证实了 CMOHEFT 的性能比较 CHEFT 并未出现下降，而 POSH 以 HEFT 得到初始解，因此也是相同的 makespan。图 3(c)表明 CMOHEFT 与 POSH 得到了相同的成本最低调度，两种算法在整个工作流的执行上均只使用了 m1.small，这主要是由该类型工作流的较低的并行度导致的，而 CHEFT 则总是代价最高的。

图 3(d)是 CMOHEFT 与 POSH 的均衡解图示，明显地，CMOHEFT 的解质量更高。具体来说，CMOHEFT 以增加约 7%的时间开销就可以平均降低若 45%的执行代价，而 POSH 得到相同的代价时，在 makespan 指标上则增加了约 25%。



(a)hypervolume (b)makespan

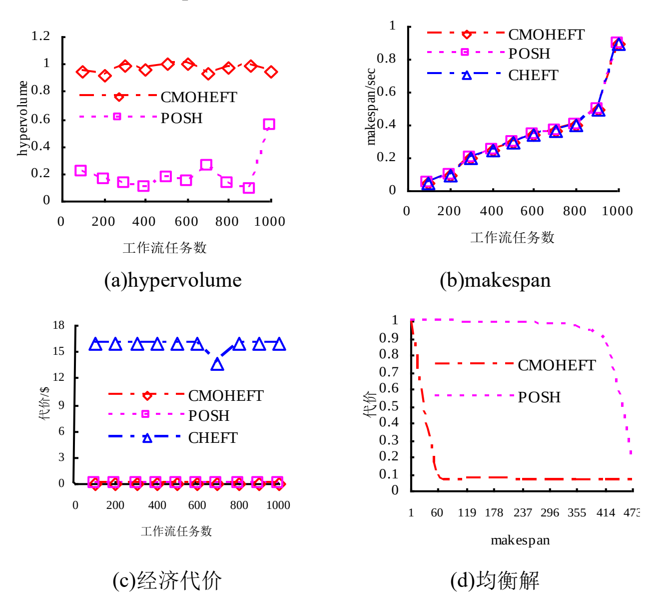
(c)经济代价 (d)均衡解

图 3 Type-1 合成工作流

2)Type-2 工作流

图 4 是 Type-2 工作流的执行结果。图 4(a)表明，对于所有工作流规模，CMOHEFT 在均衡解质量上均是优于 POSH 的。不同的工作流规模会导致拥有不同 hypervolume 值的 Pareto 边界，这表明该问题中 Pareto 边界的形状取决于可并行执行的任务数量。对于 makespan（图(b)），CMOHEFT 与 POSH 甚至在有些情况下可以得到较 CHEFT 更优的 makespan，这主要是由于 CHEFT 具有贪婪属性，较容易收敛于局部最优，而 CMOHEFT 与 POSH 以更大的搜索空间克服了这一缺陷。经济代价方面（图 4(c)），CMOHEFT 与 POSH 则以最优的 makespan 得到了相同的最低代价。

图 4(d)得到的 CMOHEFT 与 POSH 的均衡解间的不同较 Type-1 更为明显。此时，CMOHEFT 仅以增加 1.4%的 makespan 而降低了 30%左右的代价。而 POSH 得到相同的代价则需要增加 450%的 makespan 开销。



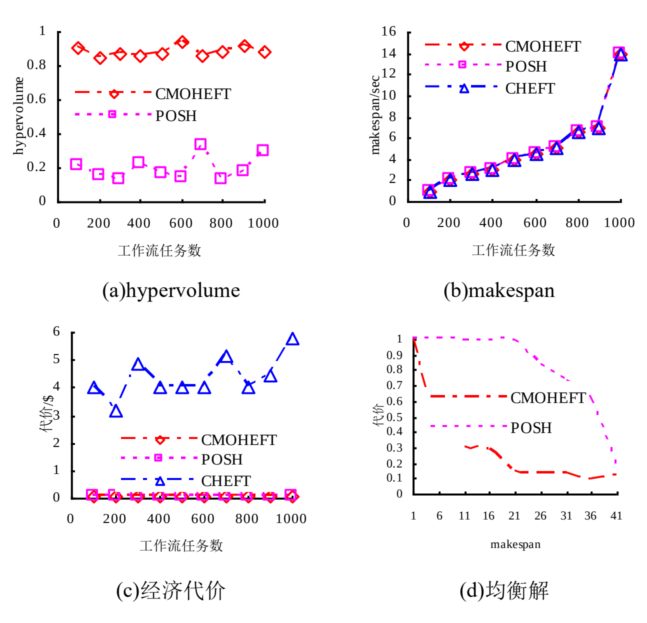
(a)hypervolume (b)makespan

(c)经济代价 (d)均衡解

图 4 Type-2 合成工作流

3)Type-3 工作流

图 5 是 Type-3 工作流的执行结果，进一步证明了以上两类工作流的验证结果。总体看来，CMOHEFT 更适应于 Amazon EC2 云环境中的工作流多目标调度。



(a)hypervolume (b)makespan

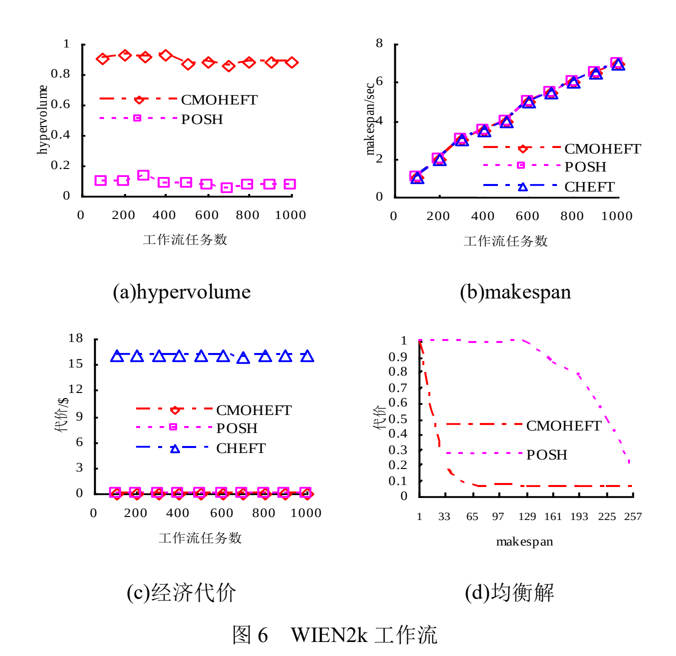
(c)经济代价 (d)均衡解

图 5 Type-3 合成工作流

6.2 现实工作流

1)WIEN2k 工作流

图 6(a)表明，与合成工作流类似，CMOHEFT 在 hypervolume 上仍优于 POSH，并且，两间的差距明显大于以上合成工作流。这是由于现实工作流应用的求解复杂度变高的原因导致的。图 6(b)和 6(c)也证实了合成工作流中的结论。在 makespan 方面，三种算法性能相同。而在代价方面，CMOHEFT 与 POSH 得到的代价更低。图 6(d)中两种算法的均衡解方面得到的结果与 Type-2 类似，CMOHEFT 可以更小的 makespan 开销节省更多的代价。



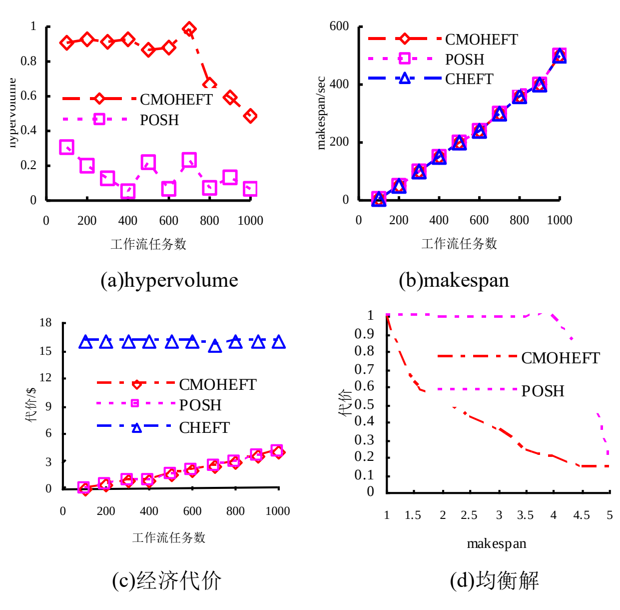
(a)hypervolume　　(b)makespan

(c)经济代价　　(d)均衡解

图 6　WIEN2k 工作流

2)POV-Ray 工作流

POV-Ray 工作流的执行结果与 WIEN2k 工作流是类似的，如图 7 所示。对于该类工作流，任务数量更高，CMOHEFT 计算高质量均衡解集合将越困难，这也导致 hypervolume 值会随着任务数量增加而下降。然而，对于 POSH 则未出现明显类似的结果，但其 Pareto 边界的质量明显也弱于 CMOHEFT。其他结果则与以上工作流类型是类似的。



(a)hypervolume　　(b)makespan

(c)经济代价　　(d)均衡解

图 7　POV-ray 工作流

6.3　资源数量的影响

本节分析放宽同步可使用资源的数量的限制对算法性能的影响，主要集中于研究资源数量规模与额外资源对计算均衡解的影响。实验配置以 WIEN2k 工作流为测试源。

图 8 和图 9 是同步资源数量设置为 20、50 和 100 时的执行结果。图 8 是资源数量对最小 makespan 的影响。可以看出，CMOHEFT 计算得到的最小 makespan 会随着资源使用数量的增加而降低。在所有情形中，算法在计算最优调度方案时均会利用所有允许使用的资源量。

图 9 是 CMOHEFH 与 POSH 计算得到的 Pareto 边界情况。此图的目的是观察不同资源数量对不同均衡解的影响。对于 CMOHEFH（图 9(a)），其 Pareto 边界在所有情况下是类似的，唯一不同仅体现在最优 makespan 的计算过程中。此时，拥有更高的资源量表明代价更高而调度时间更短（体现于边界左侧）。在最小化调度代价方面，得到的解并未受到同步使用资源量的影响。对于 POSH（图 9(b)），性能仍低于 CMOHEFT。资源量的增加会导致 POSH 的调度时间变短但代价更高。同时，其中的一些调度方案会违背允许使用的同步资源量的限制。

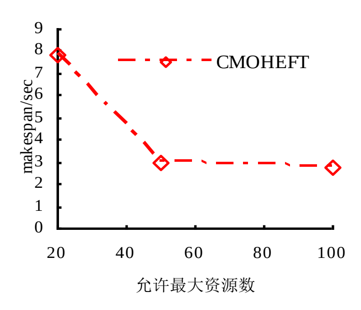
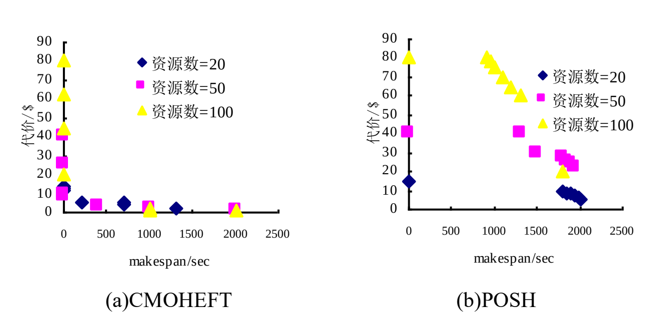


图 8　不同资源数量限制下的最短 makespan



(a)CMOHEFT　　(b)POSH

图 9　不同资源数量约束下的 Pareto 边界

7　结束语

本文提出了一种 Amazon EC2 云环境中的工作流多目标调度算法 CMOHEFT。算法以工作流任务的执行跨度 makespan 和执行经济代价的同步最小化为目标，通过启发式方法得到了多目标间的均衡最优解集合，并以 Pareto 边界和 hypervolume 值的方式对均衡解进行了有效性评估。通过三种人工合成工作流和两种现实科学工作流任务的测试，结果表明，CMOHEFT 算法比较单目标的 CHEFT 算法和多目标的 POSH 算法，大多数工作流类型中均可以得到更小的执行跨度和经济代价。

参考文献：

[1] Laili Y, Tao F, Zhang L, et, al. A study of optimal allocation of computing resources in cloud manufacturing systems [J] , International Journal of Advanced Manufacturing Technology, 2012, 63 (5): 671-690.

[2] Liu L, Zhang M, Lin Y, et al. A survey on workflow management and scheduling in cloud computing [C]// Proc of the 14th IEEE//ACM International Symposium on Cluster, Cloud and Grid Computing. [S. l.] : IEEE Press, 2014: 837-846.

[3] Rodriguez M, Buyya R. Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds [J]. IEEE Trans Cloud Computing, 2014, 2 (2): 222-235.

[4] Arabnejad H, Barbosa J G. A budget constrained scheduling algorithm for workflow applications [J]. Journal of Grid Computing, 2014, 12 (14): 1-15.

[5] Topcuoglu H, Hariri S, Wu MY, Performance-effective and low-complexity task scheduling for heterogeneous computing [J]. IEEE Trans on Parallel Distributed Systems, 2012, 13 (3): 260-274.

[6] Zheng W, Sakellariou R. Budget-deadline constrained workflow planning for admission control [J]. Journal of Grid Computing, 2013, 11 (4): 633-651.

[7] 杨玉丽，彭新光，黄名选，等，基于离散粒子群优化的云工作流调度 [J]. 计算机应用研究, 2014, 31 (12): 3677-3681.

[8] Su S, Li J, Huang Q, et. al. Cost-efficient task scheduling for executing large programs in the cloud [J]. Parallel Computing, 2013, 39 (4): 177-188.

[9] Garg R, Singh A. Multi-objective workflow grid scheduling based on discrete particle swarm optimization [M]// Swarm, Evolutionary, and Memetic Computing. Berlin: Springer, 2012: 183-190.

[10] Garg R, Singh A K. Multi-objective workflow grid scheduling using ε-fuzzy dominance sort based discrete particle swarm optimization [J]. Journal of Supercomputing, 2014, 68 (2): 709-732.

[11] Zhang F, Cao J, Hwang K, et al. Ordinal Optimized Scheduling of Scientific Workflows in Elastic Compute Clouds [C]// Proc of the 3rd IEEE International Conference on Cloud Computing Technology and Science. Washington DC: IEEE Computer Society, 2012: 9-17.

[12] Chen W, Deelman E. WorkflowSim: a toolkit for simulating scientific workflows in distributed environments [C]// Proc of the 8th IEEE International Conference on e-Science. [S. l.] : IEEE Press, 2012: 1-8.